Contents

1 Introduction 1
  1.1 Backwards Compatibility ........................................... 1
  1.2 Forwards Compatibility ............................................ 1

2 Usage Examples 2
  2.1 Simple Model ..................................................... 2
  2.2 Accessing Attachments ............................................ 3
  2.3 Inter-Scenario Data Access ....................................... 5
  2.4 Annotations ...................................................... 5

3 Core Functionality 7
  3.1 Subroutines ...................................................... 7
    insightgetappid ................................................... 8
    insightgetappname .............................................. 9
    insightgetmode ................................................ 10
    insightgetusename ............................................ 11
    insightmaximize, insightminimize ............................. 12
    insightpopulate ................................................ 13
    insightpreparectrs .......................................... 14
    insightresetprogress ......................................... 15
    insightresultsready .......................................... 16
    insightupdate ................................................ 17
    insightupdateprogress ....................................... 18
    xpri_getusername ........................................... 19

4 Annotations 20
  4.1 Entity-specific Annotations ................................... 20
  4.2 Global Annotations ............................................ 20
  4.3 List of annotations ............................................ 21

5 Repository 26
  5.1 Types ............................................................ 26
  5.2 Subroutines ...................................................... 26
    insightgetiteminfo ........................................... 27
    insightgetiteminfos ........................................ 28

6 Attachments 29
  6.1 Types ............................................................. 29
  6.2 Subroutines ...................................................... 30
    insightattachstatus .......................................... 31
    insightdeletescenattach .................................... 32
    insightgetattachbytag ....................................... 33
    insightgetprojattach ......................................... 34
    insightgetscenattach ........................................ 35
    insightlistattachtags ....................................... 36
    insightlistprojattach ....................................... 37
CHAPTER 1

Introduction

The mminsight Mosel module and package implement the interface necessary for a model to operate as part of an Xpress Insight application. The Xpress Insight Developer Guide explains how Xpress Insight interacts with the Mosel model and documents the requirements that the model needs to meet. The mminsight module provides support for a number of Xpress Insight features:

- Capturing model inputs, during a scenario ‘load’ action
- Populating model data-structures from an Xpress Insight scenario, during a scenario ‘run’ action
- Capturing model results, during a scenario ‘run’ action
- Accessing and updating scenario/project attachments
- Reading data from other Xpress Insight scenarios

This manual provides a detailed reference for each function.

A model that uses mminsight can also be run independently of Xpress Insight (from IVE or the Mosel command line) if required for debugging purposes. The XXXXX of the Developer Guide documents how to achieve this.

1.1 Backwards Compatibility

Models compiled against the current version of Xpress Insight cannot be used with earlier versions of Xpress Insight or the mminsight module. Attempting to do so will result in errors when trying to load or run a scenario.

1.2 Forwards Compatibility

It is intended that models compiled against the current version of Xpress Insight will be compatible with future versions without requiring a recompilation; please review the release notes before upgrading your Xpress Insight installation for additional information.
CHAPTER 2
Usage Examples

2.1 Simple Model

This is a version of the ‘chess2’ example, modified to be an Xpress Insight project. It demonstrates how to write a model so that Xpress Insight can capture or inject the model’s input data, depending upon the run mode, while still allowing the model developer to run the model from outside of Xpress Insight, e.g. in IVE.

```plaintext
model Chess2
  uses "mmxpr" ! Load the Xpress-Optimizer
  uses "mminsight" ! Load the Insight interface

  public declarations
    ! Model inputs
    UnitTypes: set of string
    ResourceTypes: set of string
    ResourceLimits: array(ResourceTypes) of real
    ProfitPerUnit: array(UnitTypes) of real
    UnitResourceRequirements: array(UnitTypes,ResourceTypes) of real

    ! Mathematical model objects (providing model results)
    unitstobuild: array(UnitTypes) of mpvar
    ResourceLimitConstraints: array(ResourceTypes) of linctr
    MaxProfit: linctr

  end-declarations

  ! Procedure to populate model with initial input data
  procedure loaddata
    ProfitPerUnit:: (
      "small", "large"
    ) [5, 20]
    UnitResourceRequirements:: (
      "small", "wood", "mc_time"
    ) [1, 3]
    UnitResourceRequirements:: (
      "large", "wood", "mc_time"
    ) [3, 2]
    ResourceLimits:: ("wood", "mc_time") [200, 400]
  end-procedure

  finalize( UnitTypes )
  finalize( ResourceTypes )
end-procedure

  ! Procedure to build and solve optimization problem
  procedure runproblem
    ! Build whole units only
    forall(u in UnitTypes) unitstobuild(u) is_integer

    ! Define profit
    MaxProfit:= sum(u in UnitTypes) ProfitPerUnit(u)*unitstobuild(u)

    ! Don’t use more than available resources
    forall(r in ResourceTypes)
      ResourceLimitConstraints(r) := ( sum(u in UnitTypes) UnitResourceRequirements(u,r)*unitstobuild(u) ) <= ResourceLimits(r)
end-procedure
```

Fair Isaac Corporation Confidential and Proprietary Information
2.2 Accessing Attachments

As described in the Developer Guide, Xpress Insight allows files to be attached to a project or scenario. While the model is running within Xpress Insight, the Mosel interface will allow you to query, read and edit attachments of the scenario being run, as well as query and read (but not edit) attachments of the project to which that scenario belongs.

To demonstrate the use of project and scenario attachments, in this example we have a Distances array that we want to populate with distances between cities. As this will not change, it would be very wasteful to store a copy of these in every scenario of the project, so we store them in a file distances.dat which is attached to the project. In the example, we call insightgetprojattach to copy this file to the working directory and then populate the Distances array from this. (It is assumed that the project has a companion-file that configures the Distances array as unmanaged by Xpress Insight.)

To demonstrate the use of scenario attachments, we then save the model's results to a file decorated with today's date, and attach this file to the current scenario.

```mosel
model RoutePlanner
  uses "mmxpro" ! Load the Xpress-Optimizer
  uses "mminsight" ! Load the Insight interface
  public declarations
    EvaluationDate: string
    Cities: set of string
    Distances: array(Cities,Cities) of real
```
TrafficLevels: array(Cities,Cities) of real
routetaken: array(Cities,Cities) of mpvar
end-declarations

! Procedure to build and solve optimization problem
procedure runproblem
! Download project attachment
if insightgetmode <> INSIGHT_MODE_NONE then
  ! Model is running within Insight, so download distances.dat from Insight server
  insightgetprojattach("distances.dat")
  if insightattachstatus<>INSIGHT_ATTACH_OK then
    writeln("Failed to download project attachment")
    exit(1)
  end-if
else
  ! Model is running outside of Insight, attachments are not accessible so copy from some other location
  fcopy( '/Users/me/myfiles/distances.dat', 'distances.dat' )
end-if
! Load into model
initializations from "distances.dat"
  Distances
end-initializations

! Perform optimization
buildproblem
optimizeproblem

! Capture results to file
 ROUTES_FNAME := "result-"+EvaluationDate+".dat"
 fopen(ROUTES_FNAME,F_OUTPUT)
 forall (s in Cities, d in Cities | routetaken(s,d).sol=1)
   writeln(s,"->",d) 
 fclose(F_OUTPUT)

! Store as scenario attachment
if insightgetmode <> INSIGHT_MODE_NONE then
  ! Model is running within Insight, so upload distances.dat to Insight server
  insightputscenattach(ROUTES_FNAME)
  if insightattachstatus<>INSIGHT_ATTACH_OK then
    writeln("Failed to upload scenario attachment")
    exit(1)
  end-if
end-if
end-procedure

case insightgetmode of
  INSIGHT_MODE_LOAD: do
    loaddata
  end-do
  INSIGHT_MODE_RUN: do
    insightpopulate
    runproblem
  end-do
  INSIGHT_MODE_NONE: do
    loaddata
    runproblem
  end-do
else
  writeln("Unknown execution mode")
end-case
end-model

For clarity, the implementation of the loaddata, buildproblem and optimizeproblem procedures
have been omitted from the example.
2.3 Inter-Scenario Data Access

You can also initialize your data structures with values from external scenarios using the `mminsight.scenariodata` I/O driver with a standard Mosel `initializations-from` block. For example, the following model initializes the array "Distances" from an array called "CityDrivingDistances" in a scenario "USDrivingDistances" in a project named "DrivingDistancesModel"

```mosel
model RoutePlanner
uses "mmxprs" ! Load the Xpress-Optimizer
uses "mminsight" ! Load the Insight interface

public declarations
Cities: set of string
Distances: array(Cities,Cities) of real
TrafficLevels: array(Cities,Cities) of real
routetaken: array(Cities,Cities) of mpvar
end-declarations

! Procedure to build and solve optimization problem
procedure runproblem
! Populate 'Distances' array
if insightgetmode <> INSIGHT_MODE_NONE then
! Populate 'Distances' array from the remote scenario
  initializations from "mminsight.scenariodata:/DrivingDistancesModel/USDrivingDistances"
  Distances as "CityDrivingDistances"
end-initializations
else
! Model is running outside of Insight, so populate driving distances from local file
  initializations from "/Users/me/myfiles/distances.dat"
  Distances
end-initializations
end-if

! Perform optimization
buildproblem
optimizeproblem
end-procedure

case insightgetmode of
INSIGHT_MODE_LOAD: do
  loaddata
end-do
INSIGHT_MODE_RUN: do
  insightpopulate
  runproblem
end-do
INSIGHT_MODE_NONE: do
  loaddata
  runproblem
end-do
else
  writeln("Unknown execution mode")
end-case
end-model
```

For clarity, the implementation of the `loaddata`, `buildproblem` and `optimizeproblem` procedures have been omitted from the example.

2.4 Annotations

The handling of model (schema) entities by Insight and certain global settings such as execution
modes or unit conversions are configured via annotations to the Mosel model. The following Mosel model extract shows some examples of annotation definitions.

```mosel
!@insight.manage=input
public declarations     ! All entities declared here are managed as input
!@insight.alias Suppliers
!@insight.descr Set of all suppliers
SUPP: set of string
DEP: set of string     !@insight.alias Depots
DIST: array(SUPP,DEP) of real     !@insight.unit mile
end-declarations

! The 'insight.units' or 'insight.execmodes' annotations must be stated as global
! annotations, so not immediately preceding 'declarations'

! Define a unit 'mile' with conversion rules for 'meter' and 'kilometer'

(!@insight.units.mile.
 @name mile
 @abbreviation mi
 @conversion.meter.factor=1609.34
 @conversion.kilometer.scale=0.621373
 !)

! Define an execution mode 'analyzedata'

(!@insight.execmodes.analyzedata.
 @descr Analyzing input data
 @clearinput false
 @threads 1
 !)

case insight.getmode of
 "analyzedata": do
 loaddata
 analyzedata
 end-do
INSIGHT_MODE_LOAD: do
 loaddata
 end-do
!...                          ! Handling of other execution modes
end-case
```
## CHAPTER 3

### Core Functionality

#### 3.1 Subroutines

<table>
<thead>
<tr>
<th>subroutine</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>insightgetappid</td>
<td>Query the id of the Xpress Insight application which is the parent of the model.</td>
<td>8</td>
</tr>
<tr>
<td>insightgetappname</td>
<td>Query the name of the Xpress Insight application which is the parent of the model.</td>
<td>9</td>
</tr>
<tr>
<td>insightgetmode</td>
<td>Query the Execution Mode in which Xpress Insight is running the model.</td>
<td>10</td>
</tr>
<tr>
<td>insightgetusername</td>
<td>Obtains the username of the Insight user that initiated the current scenario execution.</td>
<td>11</td>
</tr>
<tr>
<td>insightmaximize, insightminimize</td>
<td>Solves the optimization problem defined by the Mosel model.</td>
<td>55</td>
</tr>
<tr>
<td>insightpopulate</td>
<td>In the run mode INSIGHT_MODE_RUN, this procedure is called to inject the input data from the Xpress Insight scenario into the model's data structures.</td>
<td>13</td>
</tr>
<tr>
<td>insightpreparects</td>
<td>Indicate to Xpress Insight that constraints have been created and can now be relaxed or disabled, if required by the scenario inputs.</td>
<td>14</td>
</tr>
<tr>
<td>insightresetprogress</td>
<td>Resets the progress state for each progress metric back to zero.</td>
<td>56</td>
</tr>
<tr>
<td>insightresultsready</td>
<td>Indicate to Xpress Insight that the model results are available to capture by the system.</td>
<td>16</td>
</tr>
<tr>
<td>insightupdate</td>
<td>Sends a progress update notification for a single metric from the model to the Xpress Insight system.</td>
<td>58</td>
</tr>
<tr>
<td>insightupdateprogress</td>
<td>Sends update notifications for all supported progress metrics to the Xpress Insight system.</td>
<td>59</td>
</tr>
<tr>
<td>xpri_getusername</td>
<td></td>
<td>19</td>
</tr>
</tbody>
</table>
insightgetappid

**Purpose**  
Query the id of the Xpress Insight application which is the parent of the model.

**Synopsis**  
function insightgetappid:string

**Return value**  
The UID of the Xpress Insight application.
insightgetappname

**Purpose**
Query the name of the Xpress Insight application which is the parent of the model.

**Synopsis**
function insightgetappname:string

**Return value**
The name of the application.

**Further information**
The application name is not related to the name defined in the Mosel model's source code.
insightgetmode

**Purpose**
Query the Execution Mode in which Xpress Insight is running the model.

**Synopsis**
function insightgetmode:string

**Return value**
The Execution Mode, as specified in the Project. This can be a user-defined value, or can be one of these pre-defined standard values:

- INSIGHT_MODE_LOAD  When an Xpress Insight scenario is being loaded
- INSIGHT_MODE_RUN  When an Xpress Insight scenario is being run
- INSIGHT_MODE_NONE  When the model is being executed outside of Xpress Insight

**Example**

```
case insightgetmode of
  INSIGHT_MODE_LOAD: do
    ! Scenario is being 'loaded' through Xpress Insight
    ! Call user function to initialize input data and then terminate
    loaddata
    exit(0)
  end-do
  INSIGHT_MODE_RUN: do
    ! Scenario is being 'run' through Xpress Insight
    ! Populate with Insight scenario data and continue model run
    insightpopulate
  end-do
  'MY_CUSTOM_EXECUTION_MODE': do
    ! This would be an execution mode defined with an @insight.execmode
    ! annotation, to indicate some custom behavior
    insightpopulate
    ! Now take whatever custom action is appropriate...
  end-do
  else
    ! Model is being run outside of Xpress Insight
    ! Call user function to initialize input data and continue model run
    loaddata
  end-case
```

Example of the basic flow of an Insight model, calling loaddata to populate the input data entities when the scenario is loaded or the model is run from outside of Insight, and insightpopulate to populate them when scenario is run within Insight.

**Further information**
This function should be used as a conditional test to determine which parts of the model should be executed. In general, in INSIGHT_MODE_LOAD (or other user-defined execution modes intended to denote that initial data is being loaded) your model should initialize its input data and then exit. In INSIGHT_MODE_RUN (or user-defined execution modes intended to denote the model should be executed) it should call insightpopulate to populate their data structures and then construct and solve the optimization model. And in INSIGHT_MODE_NONE it should both initialize its input data and construct and solve the optimization, to allow the model developer to execute the model outside of Insight.
insightgetusername

**Purpose**
Obtains the username of the Insight user that initiated the current scenario execution.

**Synopsis**
function insightgetusername:text

**Further information**
1. When called while the model is not running within Insight, this returns ‘DEV’
2. The username returned will be the username suitable for human display - be aware that this is not a unique identifier for the user’s account, as users can change their names.
**insightmaximize, insightminimize**

**Purpose**
Solves the optimization problem defined by the Mosel model.

**Synopsis**

```plaintext
procedure insightmaximize(alg:integer, obj:linctr)
procedure insightmaximize(obj:linctr)
procedure insightminimize(alg:integer, obj:linctr)
procedure insightminimize(obj:linctr)
```

**Arguments**

- **alg**  The solver algorithm choices, as defined by mmxprs.maximize
- **obj**  The objective function constraint

**Example**

```plaintext
insightmaximize(Profit)
```

Solve the optimization model for maximum profit

```plaintext
insightminimize(XPRS_BAR,Costs)
```

Solve the optimization model to minimize costs, using the Newton-Barrier algorithm

**Further information**
These procedures should be used as a substitute for the standard mmxprs minimize/maximize procedures in an Xpress Insight compatible model. The procedures apply the disable and relaxable settings to the model before calling `minimize`/`maximize`. Before returning, the procedures capture the final runtime metrics of the solver.

**Related topics**

- `insightpreparectrs`
insightpopulate

**Purpose**
In the run mode `INSIGHT_MODE_RUN`, this procedure is called to inject the input data from the Xpress Insight scenario into the model's data structures.

**Synopsis**
procedure insightpopulate

**Example**
case insightgetmode of
  INSIGHT_MODE_LOAD: do
    ! Scenario is being 'loaded' through Xpress Insight
    ! Call user function to initialize input data and then terminate
    loaddata
    exit(0)
  end-do
  INSIGHT_MODE_RUN: do
    ! Scenario is being 'run' through Xpress Insight
    ! Populate with Insight scenario data and continue model run
    insightpopulate
    end-do
  else
    ! Model is being run outside of Xpress Insight
    ! Call user function to initialize input data and continue model run
    loaddata
  end-case

Example of the basic flow of an Insight model, calling `loaddata` to populate the input data entities when the scenario is loaded or the model is run from outside of Insight, and `insightpopulate` to populate them when scenario is run within Insight.

**Further information**
When the model is in run mode `INSIGHT_MODE_LOAD` or `INSIGHT_MODE_NONE`, calling `insightpopulate` will cause the model to abort with an error.

**Related topics**
insightgetmode
**insightpreparectrs**

**Purpose**
Indicate to Xpress Insight that constraints have been created and can now be relaxed or disabled, if required by the scenario inputs.

**Synopsis**
procedure insightpreparectrs(obj:linctr, direction:integer)

**Arguments**
direction The direction of the optimization, either INSIGHT_MINIMIZE or INSIGHT_MAXIMIZE
obj The objective function constraint

**Further information**
Constraints will be automatically relaxed or disabled if you call insightminimize or insightmaximize.

**Related topics**
insightmaximize insightminimize
insightresetprogress

**Purpose**
Resets the progress state for each progress metric back to zero.

**Synopsis**
procedure insightresetprogress

**Further information**
The Xpress Insight system reports a number of metrics during the process of solving the optimization model (see `insightupdate` and `insightupdateprogress`). This procedure sends notifications to reset the value for each metric to zero.

**Related topics**
`insightupdate insightupdateprogress`
insightresultsready

**Purpose**
Indicate to Xpress Insight that the model results are available to capture by the system.

**Synopsis**
procedure insightresultsready

**Further information**
1. This function can be used to trigger the early capture of the results data. The default behavior is for the system to capture the results data at the end of the model execution.
2. This function can be called from within a Mosel callback for example, intsol.
3. If this function is called multiple times, then only the first call will result in results data being captured by the system. Subsequent calls will be ignored.
**insightupdate**

**Purpose**
Sends a progress update notification for a single metric from the model to the Xpress Insight system.

**Synopsis**
procedure insightupdate(type:integer, value:real)

**Argument**
type The type of metric to update.
- INSIGHT_GAP The gap to the optimal solution, as a percentage
- INSIGHT_OBJVAL The best solution value found so far
- INSIGHT_NUMSOLS The count of feasible solutions found so far
- INSIGHT_OBJSENSE The direction of the solve. INSIGHT_MINIMIZE (1) for minimize, INSIGHT_MAXIMIZE (-1) for maximize.

**Example**
insightupdate(INSIGHT_OBJVAL, 51.9)

Notify Insight that the current best solution value is 51.9

**Further information**
1. This function allows the model to report back progress to the system where it is accessible by a client for display. The Analyst Client displays the progress metrics in real time in the Job Queue display.
2. The Xpress Insight event mechanism does not use the mmjobs send function.
3. By default, mminsight registers the insightupdateprogress function provided by the mminsight package as the callback function for the gapnotify callback. The insightupdateprogress function calls insightupdate for any metric that has changed. It is expected that the developer will only need to call insightupdate directly if the gapnotify callback is registered against a function provided by the developer.

**Related topics**
insightresetprogress insightupdateprogress
insightupdateprogress

**Purpose**
Sends update notifications for all supported progress metrics to the Xpress Insight system.

**Synopsis**
procedure insightupdateprogress

**Further information**
1. This procedure calls `insightupdate` for each progress metric supported.
2. By default, mminsight registers the insightupdateprogress function provided by the mminsight package as the callback function for the gapnotify callback. The insightupdateprogress function calls `insightupdate` for any metric that has changed. It is expected that the developer will only need to call `insight_update` directly if the gapnotify callback is registered against a function provided by the developer.

**Related topics**
insightupdate insightresetprogress
Core Functionality

xpri_getusername

Synopsis

    function xpri_getusername: text
Annotations defined by mminsight are either attached to specific entity declarations or apply globally to a model.

### 4.1 Entity-specific Annotations

- **insight.alias**: Used to provide an alternative name for an entity in the UI  
  - p. 21
- **insight.defaultsort**: Used to determine whether or not array groups containing this entity should be sorted by default.  
  - p. 21
- **insight.descr**: A verbose description of the entity  
  - p. 21
- **insight.format**: The formatting string used for displaying numeric values.  
  - p. 22
- **insight.hidden**: Whether to hide an entity.  
  - p. 22
- **insight.manage**: How and whether Insight handles an entity.  
  - p. 22
- **insight.nofinalize**: Prevent a set entity from being finalized during insightpopulate.  
  - p. 23
- **insight.readonly**: Whether an entity is readonly.  
  - p. 23
- **insight.transform.labels.entity**: An entity in the schema to be used as a labels entity.  
  - p. 23
- **insight.transform.labels.sortby**: How the data should be sorted.  
  - p. 24
- **insight.unit**: The base unit used to describe this entity.  
  - p. 24
- **insight.update.afterexecution**: Whether the value of the entity in the scenario is updated with the value of the corresponding model entity at the end of the scenario execution.  
  - p. 25

### 4.2 Global Annotations

- **insight.execmodes**: Category for user execution mode definitions.  
  - p. 21
- **insight.resultdata.delete**: When to delete scenario results data.  
  - p. 23
- **insight.scentypes**: Category for scenario type definitions - reserved for internal use.  
  - p. 23
- **insight.units**: Category for unit definitions.  
  - p. 24
4.3 List of annotations

The following annotations are defined by *mminsight*.

**insight**

**insight.alias**

Used to provide an alternative name for an entity in the UI.

- **Type**: text
- **Scope**: specific (must be attached to a declaration)
- **Note**: The value is used in place of the entity name where appropriate in the UI.

**insight.defaultsort**

Used to determine whether or not array groups containing this entity should be sorted by default.

- **Type**: boolean
- **Default value**: true
- **Values**: true If true then the default sort is applied, false If false then no default sort is applied
- **Scope**: specific (must be attached to a declaration)
- **Note**: If set to true then a default sort will be applied.

**insight.descr**

A verbose description of the entity.

- **Type**: text
- **Scope**: specific (must be attached to a declaration)
- **Note**: A string value providing a description of the entity

**insight.execmodes**

Category for user execution mode definitions.

**insight.execmodes.UserExecMode**

Placeholder for a user-defined execution mode name.

- **Note**: ‘UserExecMode’ must be a valid Mosel identifier (alphanumeric characters only, starting with a letter).

**insight.execmodes.UserExecMode.preferredservice**

Preferred service for this execution mode.

- **Type**: text
- **Note**: A string value specifying the preferred service to use.

**insight.execmodes.UserExecMode.clearinput**

Whether this execution mode causes data to be loaded directly by the model, rather than from the Insight server.

- **Type**: boolean
- **Note**: A boolean value indicating whether this execution mode loads data directly from the model.
**insight.execmodes.UserExecMode.descr**

A description of the execution mode

- **Type**: text
- **Note**: A string value used to provide a description for the execution mode.

**insight.execmodes.UserExecMode.threads**

Thread count available for this execution mode.

- **Type**: integer
- **Default value**: unlimited
- **Note**: An integer defining the available threads for this execution mode. If unspecified, defaults to unlimited number of threads.

**insight.format**

The formatting string used for displaying numeric values.

- **Type**: text
- **Scope**: specific (must be attached to a declaration)
- **Note**: The string value is taken as a formatting string.

**insight.hidden**

Whether to hide an entity.

- **Type**: text
- **Default value**: false
- **Values**: always indicates that the UI should hide the entity always, false indicates that the UI should show the entity where appropriate, true indicates that the UI should hide the entity where appropriate
- **Scope**: specific (must be attached to a declaration)
- **Note**: If true, indicates that the UI should hide the entity where appropriate.
- **See also**: insight.manage, insight.readonly

**insight.manage**

How and whether Insight handles an entity.

- **Type**: text
- **Default value**: default
- **Values**: default Entity is included in the input data if the model declares it as of basic type, and is included in the result data if the model declares it as of mp-type. ignore Excluded from the schema input Included in the scenario input data. result Included in the scenario results data
- **Scope**: specific (must be attached to a declaration)
- **Note**: Defines how the system manages the entity data.
- **See also**: insight.hidden
**insight.nofinalize**

Prevent a set entity from being finalized during insightpopulate.

- **Type**: text
- **Scope**: specific (must be attached to a declaration)
- **Note**: Specifies that the set entity should not be finalized during insightpopulate, regardless of the entity’s properties in the Insight schema. This would usually be used if you needed to add additional values to a previously-finalized set following a model upgrade.

**See also** insight.nofinalize

**insight.readonly**

Whether an entity is readonly.

- **Type**: boolean
- **Default value**: false
- **Scope**: specific (must be attached to a declaration)
- **Note**: Specifies that the value(s) of the entity cannot be modified.

**See also** insight.hidden

**insight.resultdata.delete**

When to delete scenario results data.

- **Type**: text
- **Default value**: on-change
- **Values**:
  - on-change: Delete scenario result data when the scenario input-data is edited, or when scenario is queued for execution.
  - on-execute: Delete scenario result data when scenario starts to execute.
  - on-queue: Delete scenario result data when scenario is queued for execution.
- **Scope**: global (not attached to any declaration)
- **Note**: Results data is deleted when a certain state change occurs for the scenario. This attribute identifies this state change as either whenever a scenario is modified, when it is queued, or when it begins execution.

**See also** insight.manage

**insight.scentypes**

Category for scenario type definitions - reserved for internal use.

**insight.transform.labels.entity**

An entity in the schema to be used as a labels entity.

- **Type**: text
- **Scope**: specific (must be attached to a declaration)
- **Note**: The value is the name of the entity. The type of the index set of the labels entity much match the data type of this entity. The data type of the labels entity can be any primitive type.
insight.transform.labels.sortby
How the data should be sorted.
Type text
Values
label The data will be sorted by label
value The data will be sorted by value
Scope specific (must be attached to a declaration)
Note The value can be either value or label to sort by either the underlying value or the label.

insight.unit
The base unit used to describe this entity.
Type text
Scope specific (must be attached to a declaration)
Note The value is the unit used to describe the entity.

insight.units
Category for unit definitions.

insight.units.UserUnitId
Placeholder for a user-defined unit name.
Note ‘UserUnitId’ must be a valid Mosel identifier (alphanumeric characters only, starting with a letter).

insight.units.UserUnitId.abbreviation
The abbreviation of the unit.
Type text
Note A string value specifying an abbreviation for the unit.

insight.units.UserUnitId.name
The textual description of the unit.
Type text
Note A string value specifying the name of the unit.

insight.units.UserUnitId.conversion
Category for unit conversion definitions.

insight.units.UserUnitId.conversion.UserUnitId2
Placeholder for a user-defined unit name.
Note ‘UserUnitId2’ must be a valid Mosel identifier (alphanumeric characters only, starting with a letter).

insight.units.UserUnitId.conversion.UserUnitId2.factorentity
The factor entity used to convert the unit. This is an entity reference.
Type text
Note String value specifying the entity name to be used for converting the unit ‘UserUnitId’ to ‘UserUnitId2’.
**insight.units.UserUnitId.conversion.UserUnitId2.factor**

The factor used to convert the unit. This is an absolute value.

*Type*  
real

*Note*  
Double value specifying the factor to be applied for converting the unit ‘UserUnitId’ to ‘UserUnitId2’.

**insight.units.UserUnitId.conversion.UserUnitId2.offset**

Optional offset value used to convert the unit.

*Type*  
real

*Default value*  
0

*Note*  
Double value specifying the offset to be used when converting the unit ‘UserUnitId’ to ‘UserUnitId2’.

**insight.units.UserUnitId.conversion.UserUnitId2.scale**

Optional scale value used to convert the unit.

*Type*  
real

*Default value*  
1

*Note*  
Double value specifying any scaling to be applied for converting the unit ‘UserUnitId’ to ‘UserUnitId2’.

**insight.update.afterexecution**

Whether the value of the entity in the scenario is updated with the value of the corresponding model entity at the end of the scenario execution.

*Type*  
boolean

*Default value*  
false

*Scope*  
specific (must be attached to a declaration)

*Note*  
If true the value of the entity is updated to correspond with the model entity after execution.
CHAPTER 5
Repository

5.1 Types

\textit{insightiteminfo} : \texttt{record}

\begin{itemize}
\item \texttt{id} : \texttt{text}
  \hspace{1cm} item id
\item \texttt{type} : \texttt{text}
  \hspace{1cm} item type (FOLDER or scenario type)
\item \texttt{name} : \texttt{text}
  \hspace{1cm} item name
\item \texttt{path} : \texttt{text}
  \hspace{1cm} item path
\item \texttt{parentpath} : \texttt{text}
  \hspace{1cm} item parent path
\end{itemize}

5.2 Subroutines

\begin{itemize}
\item \texttt{insightgetiteminfo} \hspace{1cm} Get information for a repository item with the supplied path \hspace{1cm} p. 27
\item \texttt{insightgetiteminfos} \hspace{1cm} Get information for items in the folder with the supplied path \hspace{1cm} p. 28
\end{itemize}
insightgetiteminfo

**Purpose**
Get information for a repository item with the supplied path

**Synopsis**
function insightgetiteminfo(path:text):insightiteminfo

**Return value**
Information about the repository item (scenario / folder)

**Example**
```plaintext
declarations
    info: insightiteminfo
end-declarations
info := insightgetiteminfo('/mymodel/myscenario')
```

Example of using insightgetiteminfo to obtain info for a scenario

```plaintext
declarations
    info: insightiteminfo
end-declarations
info := insightgetiteminfo('.')
```

Example of using insightgetiteminfo to obtain info for the current scenario

**Further information**
1. Attempting to get repository item info when the model is not being run through Xpress Insight will cause the model to abort with an error.
2. Attempting to get repository item info when the model is not being run through Xpress Insight will cause the model to abort with an error.
insightgetiteminfos

**Purpose**
Get information for items in the folder with the supplied path

**Synopsis**
function insightgetiteminfos(folderpath:text):list of insightiteminfo

**Return value**
Information about the items (folders / scenarios). This does not return information about Virtual Scenario Groups.

**Example**

declarations
  info: list of insightiteminfo
end-declarations
info := insightgetiteminfos('/mymodel/myfolder')

Example of using insightgetiteminfo to obtain info for items in a folder

declarations
  info: list of insightiteminfo
end-declarations
info := insightgetiteminfos('.')

Example of using insightgetiteminfo to obtain info for items the same folder as the current scenario

**Further information**
Attempting to get repository item info when the model is not being run through Xpress Insight will cause the model to abort with an error.
CHAPTER 6
Attachments

6.1 Types

**insightattachment** : record
- a record containing information about a single attachment
  - filename : text
    - filename of the attachment
  - description : text
    - description of the attachment
  - tags : list of string
    - collection of tags associated with the attachment
  - size : integer
    - size of the attachment, in bytes
  - lastModifiedUser : text
    - name of the last Insight user to modify the attachment
  - lastModifiedDate : datetime
    - date and time of last modification to attachment
  - hidden : boolean
    - whether the attachment is hidden from the UI

**Note** Modifying an insightattachment record will not modify the attachment information on the server.

**insightattachmenttag** : record
- a record containing information about a tag defined in the project's companion file
  - name : string
    - name of the tag
  - description : string
    - description of the tag
  - mandatory : boolean
    - whether the tag is mandatory
  - usage : string
    - tag usage restrictions, either 'single-file' or 'multi-file'
  - mandatory : boolean
Note

Modifying an insightattachmenttag record will not modify the attachment tag information on the server.

6.2 Subroutines

insightattachstatus  Indicates the status of the most recent attempt to access or modify an attachment.  p. 31

insightdeletescenattach  Deletes a scenario attachment.  p. 32

insightgetattachbytag  Searches the scenario and the containing project for an attachment or attachments with the given tag, and retrieves them from the Insight server, placing them in the Mosel working directory where they can be read by the model. If any scenario attachments with the given tag are found, these are retrieved without searching the project. If no scenario attachments with the given tag are found, then the search continues at the project level.  p. 33

insightgetprojattach  Retrieves a project attachment from the Insight server, placing it in the Mosel working directory where it can be read by the model.  p. 34

insightgetscenattach  Retrieves an attachment from the Insight server either for a given scenario, placing it in the Mosel working directory where it can be read by the model.  p. 35

insightlistattachtags  Retrieves a list of the attachment tags defined in the companion file p. 36

insightlistprojattach  Retrieves a list of all the files attached to the project.  p. 37

insightlistprojattachbytag  Retrieves a list of all the files attached to the project with the given tag.  p. 38

insightlistsценattach  Retrieves a list of all the files attached to a given scenario.  p. 39

insightlistsценattachbytag  Retrieves a list of all the files attached to a scenario with the given tag.  p. 40

insightprojattachinfo  Retrieves information about a given project attachment  p. 41

insightputscenattach  Uploads a scenario attachment to the Insight server, reading it from the Mosel working directory.  p. 42

insightrenamescenattach  Renames an existing scenario attachment.  p. 43

insightsценattachinfo  Retrieves information about a given scenario attachment  p. 44

insightsetscenattachdesc  Update the description of an existing scenario attachment.  p. 45

insightsetscenattachhidden  Mark an existing scenario attachment as hidden or visible in the Xpress Insight UI.  p. 46

insightsetscenattachtags  Update the tags of an existing scenario attachment.  p. 47
### insightattachstatus

**Purpose**
Indicates the status of the most recent attempt to access or modify an attachment.

**Synopsis**

```plaintext
function insightattachstatus:integer
```

**Return value**
One of the following constants:

- **INSIGHT_ATTACH_OK** The operation completed successfully.
- **INSIGHT_ATTACH_RUNTIME_ERROR** An internal error occurred.
- **INSIGHT_ATTACH_NOT_FOUND** The specified attachment does not exist.
- **INSIGHT_ATTACH_SEVERAL_FOUND** Several attachments match the given tag but the procedure only allows for one to be retrieved.
- **INSIGHT_ATTACH_INVALID_FILENAME** An attachment could not be created or renamed because the specified filename is invalid. It may be too long, too short, or contain invalid characters.
- **INSIGHT_ATTACH_INVALID_DESCRIPTION** The specified description is invalid. The description can be a maximum of 2500 characters in length.
- **INSIGHT_ATTACH_ALREADY_EXISTS** An attachment could not be created because another attachment with the same name already exists.
- **INSIGHT_ATTACH_TOO_LARGE** An attachment could not be created because another attachment with the same name already exists. Attachments can be a maximum of 60Mb in size.
- **INSIGHT_ATTACH_TOO_MANY** An attachment could not be created because the maximum number of attachments (100) has been reached for the project or scenario.

**Further information**
After every call to an attachment-related function or procedure, you should check the value of `insightattachstatus` to see if your request succeeded.

**Related topics**

- `insightgetscenattach`
- `insightlists scenattach`
- `insightlistscenattachbytag`
- `insightputscenattach`
- `insightrenamescenattach`
- `insightscenattachinfo`
- `insightsetscenattachdesc`
- `insightsetscenattachhidden`
- `insightsetscenattachtags`
- `insightgetprojattach`
- `insightlistprojattach`
- `insightlistprojattachbytag`
- `insightprojattachinfo`
**insightdeletescenattach**

**Purpose**
 Deletes a scenario attachment.

**Synopsis**

```plaintext
procedure insightdeletescenattach(filename:text)
```

**Argument**

- `filename`  
  The filename of the attachment to be deleted.

**Example**

```plaintext
insightdeletescenattach('myattach.dat')
if insightattachstatus=INSIGHT_ATTACH_OK then
  writeln("Attachment deleted")
else
  writeln("Error deleting attachment")
end-if
```

Example of deleting an attachment called `myattach.dat` from the current scenario.

**Further information**

1. Check the attachment status code using `insightattachstatus` to determine whether the attachment will be successfully deleted.
2. The attachment will still be available on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**

- `insightattachstatus`
insightgetattachbytag

Purpose
Searches the scenario and the containing project for an attachment or attachments with the given tag, and retrieves them from the Insight server, placing them in the Mosel working directory where they can be read by the model. If any scenario attachments with the given tag are found, these are retrieved without searching the project. If no scenario attachments with the given tag are found, then the search continues at the project level.

Synopsis
procedure insightgetattachbytag(tag:text, attachments:list of insightattachment)
procedure insightgetattachbytag(tag:text, attachment:insightattachment)
procedure insightgetattachbytag(tag:text, filenames:list of text)

Arguments
- **tag**: The tag to search for
- **attachment**: An attachment object which will be populated with the details of the attachment that was retrieved
- **attachments**: A list which will be populated with the details of the attachments that were retrieved
- **filenames**: A list which will be populated with the filenames of the attachments that were retrieved

Example
```mosel
declarations
  attachment: insightattachment
end-declarations
insightgetattachbytag('mytag1', attachment)
if insightattachstatus<>INSIGHT_ATTACH_OK then
  writeln("Error searching for attachments")
else
  initializations from attachment.filename
  MY_ARRAY
  end-initializations
end-if
```

Example of searching for and retrieving an attachment with the tag *tag1*

Further information
1. Check the attachment status code using `insightattachstatus` to determine whether the attachment(s) were successfully retrieved.
2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

Related topics
- `insightattachstatus`
insightgetprojattach

**Purpose**
Retrieves a project attachment from the Insight server, placing it in the Mosel working directory where it can be read by the model.

**Synopsis**
procedure insightgetprojattach(filename:text)

**Argument**
filename The filename of the attachment to be retrieved.

**Example**

```plaintext
insightgetprojattach('myattach.dat')
if insightattachstatus=INSIGHT_ATTACH_OK then
  initializations from 'myattach.dat'
  MY_ARRAY
end-initializations
else
  writeln("Error retrieving attachment")
end-if
```

Example of copying a project attachment called myattach.dat to the working directory.

**Further information**

1. Check the attachment status code using `insightattachstatus` to determine whether the attachment was successfully fetched.

2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**

- `insightattachstatus`
insightgetscenattach

**Purpose**
Retrieves an attachment from the Insight server either for a given scenario, placing it in the Mosel working directory where it can be read by the model.

**Synopsis**

```plaintext
procedure insightgetscenattach(filename:text)
procedure insightgetscenattach(filename:text, scenariopath:text)
```

**Arguments**

- `filename`: The filename of the attachment to be retrieved.
- `scenariopath`: The path of a scenario. A scenario path is the full path to a scenario name starting from the repository root and including the project name. E.g. `/myproject/FolderA/FolderB/myscenario` If the scenario path is not specified, the attachment is retrieved for the current scenario.

**Example**

Example of copying a scenario attachment called `myattach.dat` to the working directory.

```plaintext
insightgetscenattach('myattach.dat')
if insightattachstatus=INSIGHT_ATTACH_OK then
    initializations from 'myattach.dat'
    MY_ARRAY
    end-initializations
else
    writeln("Error retrieving attachment")
end-if
```

Getting an attachment for the current scenario.

```plaintext
insightgetscenattach('myattach.dat', '/myproject/FolderA/FolderB/myscenario')
if insightattachstatus=INSIGHT_ATTACH_OK then
    initializations from 'myattach.dat'
    MY_ARRAY
    end-initializations
else
    writeln("Error retrieving attachment")
end-if
```

Getting an attachment for a scenario with path `/myproject/FolderA/FolderB/myscenario`.

**Further information**

1. Check the attachment status code using `insightattachstatus` to determine whether the attachment was successfully fetched.
2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**

- `insightattachstatus`
- `insightputscenattach`
insightlistattachtags

**Purpose**
Retrieves a list of the attachment tags defined in the companion file

**Synopsis**
function insightlistattachtags: list of insightattachmenttag

**Return value**
A list of the defined tags

**Example**
```
tagslist := insightlistattachtags
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Defined tags: ",tagslist)
else
    writeln("Error retrieving tags list")
end-if
```

Example of outputting list of tags defined in companion file

**Further information**
1. Check the attachment status code using `insightattachstatus` to determine whether the attachment was successfully fetched.
2. Attempting to access attachment tags when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- `insightsetscenattachtags`
insightlistprojattach

**Purpose**
Retrieves a list of all the files attached to the project.

**Synopsis**
function insightlistprojattach:list of insightattachment

**Return value**
A list of the project attachments

**Example**
```plaintext
declarations
    myattachments: list of insightattachment
end-declarations
myattachments := insightlistprojattach
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachments: ", myattachments)
else
    writeln("Error listing attachments")
end-if
```

Example of fetching information about all attachments on the project containing the current scenario into a list called *myattachments*

**Further information**
1. Check the attachment status code using `insightattachstatus` to determine whether the attachments list was successfully retrieved.
2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- `insightattachstatus`
- `insightgetprojattach`
- `insightlistprojattachbytag`
insightlistprojattachbytag

Purpose
Retrieves a list of all the files attached to the project with the given tag.

Synopsis
function insightlistprojattachbytag(tag:text):list of insightattachment

Argument
tag        The tag to search for

Return value
A list of the project attachments

Example

declarations
    myattachments: list of insightattachment
end-declarations
myattachments := insightlistprojattachbytag('mytag1')
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachments: ", myattachments)
else
    writeln("Error listing attachments")
end-if

Example of fetching information about all attachments on the project with the tag tag1 into a list called myattachments

Further information

1. Check the attachment status code using insightattachstatus to determine whether the attachments list was successfully retrieved.

2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

Related topics
insightattachstatus insightgetprojattach insightlistprojattach
### insightlistsccnattach

**Purpose**
Retrieves a list of all the files attached to a given scenario.

**Synopsis**
```plaintext
function insightlistsccnattach: list of insightattachment
function insightlistsccnattach(scenariopath: text): list of insightattachment
```

**Argument**
- **scenariopath**
  The path of a scenario. A scenario path is the full path to a scenario name starting from the repository root and including the project name. E.g. `/myproject/FolderA/FolderB/myscenario` If the scenario path is not specified, the attachment is retrieved for the current scenario.

**Return value**
A list of the scenario attachments.

**Example**
Example of fetching information about all attachments of a scenario into a list called *myattachments*
```
declarations
  myattachments: list of insightattachment
end-declarations
myattachments := insightlistsccnattach
if insightattachstatus=INSIGHT_ATTACH_OK then
  writeln("Attachments: ", myattachments)
else
  writeln("Error listing attachments")
end-if
```

Getting the list of attachments for the current scenario
```
declarations
  myattachments: list of insightattachment
end-declarations
myattachments := insightlistsccnattach('/myproject/FolderA/FolderB/myscenario')
if insightattachstatus=INSIGHT_ATTACH_OK then
  writeln("Attachments: ", myattachments)
else
  writeln("Error listing attachments")
end-if
```

Getting the list of attachments for a scenario with path `/myproject/FolderA/FolderB/myscenario`

**Further information**
1. Check the attachment status code using *insightattachstatus* to determine whether the attachments list was successfully retrieved.
2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- insightattachstatus
- insightgetscenattach
- insightlistsccnattachbytag
insightlistscenattachbytag

**Purpose**
Retrieves a list of all the files attached to a scenario with the given tag.

**Synopsis**
function insightlistscenattachbytag(tag:text):list of insightattachment
function insightlistscenattachbytag(tag:text, scenariopath:text):list of insightattachment

**Arguments**
tag The tag to search for
scenariopath The path of a scenario. A scenario path is the full path to a scenario name starting from the repository root and including the project name. E.g. /myproject/FolderA/FolderB/myscenario If the scenario path is not specified, the attachment is retrieved for the current scenario

**Return value**
A list of the scenario attachments

**Example**
Example of fetching information about all attachments on a scenario with the tag tag1 into a list called myattachments

```plaintext
declarations
    myattachments: list of insightattachment
end-declarations
myattachments := insightlistscenattachbytag('mytag1')
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachments: ", myattachments)
else
    writeln("Error listing attachments")
end-if
```

Getting the list of attachments for the current scenario with the given tag.

```plaintext
declarations
    myattachments: list of insightattachment
end-declarations
myattachments := insightlistscenattachbytag('mytag1', '/myproject/FolderA/FolderB/myscenario')
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachments: ", myattachments)
else
    writeln("Error listing attachments")
end-if
```

Getting the list of attachments with the given tag for a scenario with path /myproject/FolderA/FolderB/myscenario.

**Further information**
1. Check the attachment status code using insightattachstatus to determine whether the attachments list was successfully retrieved.
2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
insightattachstatus insightgetscenattach insightlistscenattach
insightprojattachinfo

**Purpose**
Retrieves information about a given project attachment

**Synopsis**
function insightprojattachinfo(filename:text):insightattachment

**Argument**
filename    The filename of the project attachment to request

**Return value**
Information about the attachment

**Example**

```plaintext
declarations
    myattachment: insightattachment
end-declarations
myattachment := insightprojattachinfo('myattach.dat')
if insightattachstatus=INSIGHT.Attach_OK then
    writeln("Attachment description: ", myattachment.description)
else
    writeln("Error querying attachment")
end-if
```

Example of copying information about the attachment `myattach.dat` on the project containing the current scenario into a record called `myattachment`

**Further information**

1. Check the attachment status code using `insightattachstatus` to determine whether the attachments information was successfully retrieved.

2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
`insightattachstatus`
insightputscenattach

**Purpose**
Uploads a scenario attachment to the Insight server, reading it from the Mosel working directory.

**Synopsis**
procedure insightputscenattach(filename:text, overwrite:boolean)
procedure insightputscenattach(filename:text)

**Arguments**
- filename: The filename of the attachment to be uploaded
- overwrite: If true, will overwrite attachment if it already exists. If false an attachment already exists, will fail with insightattachstatus INSIGHT_ATTACH_ALREADY_EXISTS. Defaults to true if not given.

**Example**

```plaintext
insightputscenattach('myattach.dat',false)
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachment added ok")
elif insightattachstatus=INSIGHT_ATTACH_ALREADY_EXISTS then
    writeln("Attachment already exists")
else
    writeln("Error adding attachment")
end-if
```

Example of taking a file `myattach.dat` in the working directory, and saving it as a new scenario attachment called `myattach.dat`.

**Further information**
1. Check the attachment status code using `insightattachstatus` to determine whether the attachment was successfully added.
2. The new attachment will not be available on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- `insightattachstatus`
- `insightgetscenattach`
**insightrenamescenattach**

**Purpose**
Renames an existing scenario attachment.

**Synopsis**
```plaintext
procedure insightrenamescenattach(oldname:text, newname:text)
```

**Arguments**
- `oldname`: The existing filename of the attachment to be renamed
- `newname`: The new filename of the attachment. Must not already be used for a scenario attachment.

**Example**
```plaintext
insightrenamescenattach('myattach.dat','myattach-2015.dat')
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachment renamed ok")
else
    writeln("Error renaming attachment")
end-if
```

Example of renaming an existing attachment of the current scenario from `myattach.dat` to `myattach-2015.dat`.

**Further information**
1. Check the attachment status code using `insightattachstatus` to determine whether the new attachment name was accepted.
2. The attachment will not be renamed on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- `insightattachstatus`
insightscenattachinfo

**Purpose**  
Retrieves information about a given scenario attachment

**Synopsis**  
function insightscenattachinfo(filename:text):insightattachment

**Argument**  
filename  
The filename of the scenario attachment to request

**Return value**  
Information about the attachment

**Example**

```plaintext
declarations
   myattachment: insightattachment
end-declarations
myattachment := insightscenattachinfo('myattach.dat')
if insightattachstatus=INSIGHT_ATTACH_OK then
   writeln("Attachment description: ", myattachment.description)
else
   writeln("Error querying attachment")
end-if
```

Example of copying information about the attachment `myattach.dat` on the current scenario into a record called `myattachment`

**Further information**

1. Check the attachment status code using `insightattachstatus` to determine whether the attachments information was successfully retrieved.

2. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**

- `insightattachstatus`
- `insightsetscenattachdesc`
insightsetscenattachdesc

Purpose
Update the description of an existing scenario attachment.

Synopsis
procedure insightsetscenattachdesc(filename:text, description:text)

Arguments
filename The filename of the scenario attachment to update
description The new description of the attachment

Example
insightsetscenattachdesc('myattach.dat','This is my first attachment')
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachment description updated ok")
else
    writeln("Error updating attachment")
end-if

Example of setting the description of a scenario attachment myattach.dat to be "This is my first attachment"

Further information
1. Check the attachment status code using insightattachstatus to determine whether the new attachment description was accepted.
2. The attachment will not be updated on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

Related topics
insightattachstatus
**insightsetscenattachhidden**

**Purpose**
Mark an existing scenario attachment as hidden or visible in the Xpress Insight UI.

**Synopsis**
procedure insightsetscenattachhidden(filename:text, hidden:boolean)

**Arguments**
- filename: The filename of the scenario attachment to hide or show
- hidden: If true, the attachment will be hidden in the Xpress Insight UI; if false, it will be visible

**Example**

```
insightsetscenattachhidden('myattach.dat',true)
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachment hidden ok")
else
    writeln("Error hiding attachment")
end-if
```

Example of hiding of a scenario attachment myattach.dat

**Further information**
1. Check the attachment status code using `insightattachstatus` to determine whether the update was successful.
2. The attachment will not be updated on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.

**Related topics**
- `insightattachstatus`
insightsetscenattachtags

Purpose
Update the tags of an existing scenario attachment.

Synopsis
procedure insightsetscenattachtags(filename:text, tags:list of string)

Arguments
filename The filename of the scenario attachment to update
tags The new tags to apply to the attachment. Any existing tags will be removed.

Example
insightsetscenattachtags('myattach.dat', ['mytag1','mytag2'])
if insightattachstatus=INSIGHT_ATTACH_OK then
    writeln("Attachment tags updated ok")
else
    writeln("Error updating attachment")
end-if

Example of setting the tags of a scenario attachment myattach.dat to be "mytag1" and "mytag2"

Further information
1. Check the attachment status code using insightattachstatus to determine whether the new attachment tags were accepted.
2. The attachment will not be updated on the Insight server until the scenario completes.
3. Attempting to access attachments when the model is not being run through Xpress Insight will cause the model to abort with an error.
4. If any of the specified tags are single-file tags, they will be removed from other scenarios as part of this operation.

Related topics
insightattachstatus
CHAPTER 7
Inter-Scenario Data Access

7.1 I/O drivers

7.1.1 Driver scenariodata

The driver can only be used in ‘initializations from’ blocks. The SCENARIO_REPOSITORY_PATH must be the repository path (of the form "/ProjectName/FolderName/ScenarioName") of the scenario from which data will be read. Within the initializations-from block you should specify the name of the local entity to populate, and the name of the external entity from which it will be populated, ie:

    initializations from "mminsight.scenariodata:<repository path>"
    LOCAL ENTITY_NAME_1 as "ENTITY_NAME_1_IN_EXTERNAL_SCENARIO"
    LOCAL ENTITY_NAME_2 as "ENTITY_NAME_2_IN_EXTERNAL_SCENARIO"
    LOCAL ENTITY_NAME_3 as "ENTITY_NAME_3_IN_EXTERNAL_SCENARIO"
    end-initializations

Both input and result entities can be read from the external scenario. There is no limitation on the number of external entities you can specify. To access more than one scenario, use multiple initializations from blocks.

If the external scenario does not exist, or is not readable by the user executing the scenario, the model will abort with an error.

When the model is run from outside of Xpress Insight, attempting to use the scenariodata I/O driver will cause the model to abort with an error.
# Parameters

The `mminsight` module provides several parameters that can be used to read information from the Insight server or change behaviour of the scenario execution. These can be accessed using the standard `getparam` and `setparam` Mosel functions, or set with the "mminsight." prefix through the Parameters array in Insight.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>insight_job_input</code></td>
<td>Job description file</td>
<td>mminsight</td>
<td>49</td>
</tr>
<tr>
<td><code>insight_nofinalize</code></td>
<td>Prevent finalization during <code>insightpopulate</code></td>
<td></td>
<td>49</td>
</tr>
<tr>
<td><code>insight_scenario_id</code></td>
<td>Scenario UID</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td><code>insight_scenario_name</code></td>
<td>Scenario Name</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td><code>insight_scenario_path</code></td>
<td>Scenario Path</td>
<td></td>
<td>50</td>
</tr>
<tr>
<td><code>insight_verbose</code></td>
<td>Additional Logging Level</td>
<td></td>
<td>51</td>
</tr>
<tr>
<td><code>insight_writeprob</code></td>
<td>Capture Optimization Problem</td>
<td></td>
<td>51</td>
</tr>
</tbody>
</table>

### `insight_job_input`

**Description**: It specifies the source of the input data to be loaded when executing a scenario. When debugging an Xpress Insight model outside of Xpress Insight, e.g. in IVE, the developer can use this parameter to specify the location of a job file that was previously exported from the Xpress Insight system.

**Type**: String, read/write

**Default value**: Set by Xpress Insight

**Note**: This parameter is used by Xpress Insight to pass scenario information to the model. If the user sets this within Xpress Insight, the scenario will not run.

**Module**: mminsight

### `insight_nofinalize`

**Description**: Allows developer to prevent the finalization of any sets by the ‘`insightpopulate`’ call

**Type**: Boolean, read/write
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Default value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>false</td>
<td>false</td>
<td>insightpopulate will try to finalize sets that were finalized when the first scenario in the app was loaded.</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td>insightpopulate will not finalize any index sets</td>
</tr>
</tbody>
</table>

### Notes

1. If you set this to ‘true’, your model must take responsibility for finalizing any sets that require finalization.
2. Not finalizing index sets will prevent cells in static arrays from being automatically created on first access.
3. Not finalizing index sets can have performance implications when a scenario contains large arrays.

### Affects routines

- insightpopulate

### Module

- mminsight

---

### insight_scenario_id

<table>
<thead>
<tr>
<th>Description</th>
<th>The globally unique identifier for the scenario currently being executed by the model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>String, read only</td>
</tr>
<tr>
<td>Default value</td>
<td>UID of scenario</td>
</tr>
<tr>
<td>Note</td>
<td>This parameter is read-only.</td>
</tr>
</tbody>
</table>

### Module

- mminsight

---

### insight_scenario_name

<table>
<thead>
<tr>
<th>Description</th>
<th>The name of the scenario currently being executed by the model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>String, read only</td>
</tr>
<tr>
<td>Default value</td>
<td>Name of scenario</td>
</tr>
<tr>
<td>Note</td>
<td>This parameter is read-only.</td>
</tr>
</tbody>
</table>

### Module

- mminsight

---

### insight_scenario_path

<table>
<thead>
<tr>
<th>Description</th>
<th>The repository path of the scenario currently being executed by the model.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>String, read only</td>
</tr>
<tr>
<td>Default value</td>
<td>Repository path of scenario, e.g. &quot;/ProjectName/FolderName/ScenarioName&quot;</td>
</tr>
<tr>
<td>Note</td>
<td>This parameter is read-only.</td>
</tr>
</tbody>
</table>

### Module

- mminsight

### insight_verbose

**Description**
Enables extra logging messages to be sent to the scenario’s run log when it is set to a non-zero value.

**Type**
Integer, read/write

**Values**
Between 0 and 100, as follows:

- 0: No logging
- 1: Basic logging of the steps undertaken by mminsight
- 2: As 1, but with additional logging of which entities are being captured/injected/etc
- 3: As 2, but log also when mminsight finalizes a set
- 5: As 3, but with additional logging of how sets/arrays are being stored
- 6+: Values between 6 and 100 turn on debugging output and should only be used on the advice of FICO support.

**Default value**
0

**Affects routines**
insightpopulate, insightminimize, insightmaximize

**Module**
mminsight

### insight_writeprob

**Description**
When set to a non-empty string, mminsight will call mmxprs.writeprob(insight_writeprob, "p") immediately prior to solving the optimization problem.

**Type**
String, read/write

**Values**
Any string, representing filename to which to write exported problem

**Default value**
Empty string

**Note**
Can be used to capture Xpress optimization problem being executed by Insight, in order to tune or debug optimization outside of Xpress Insight.

**Affects routines**
insightminimize, insightmaximize

**Module**
mminsight
Chapter 9

Deprecated Functionality

The functionality described in this section has been deprecated; while models that use the deprecated parameters and functions are still supported, we do not recommend their continued use.

9.1 Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Default value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>insight_model_id</td>
<td>The globally unique identifier for the project parent of the model. This parameter is read-only. Replaced with <code>insightgetappid</code>.</td>
<td>String, read only</td>
<td>UID of project</td>
<td>1. The <code>insight_model_name</code> is not related to the name defined in the Mosel model's source code.</td>
</tr>
<tr>
<td>insight_model_name</td>
<td>The name of the project as specified by the companion file or the user. Replaced with <code>insightgetappname</code>.</td>
<td>String, read only</td>
<td>project name</td>
<td>2. This parameter is read-only.</td>
</tr>
</tbody>
</table>
9.2 Subroutines

- `insight_end_initializations`: Indicates that initializations are completed, p. 54
- `insight_minimize, insight_maximize`: Solves the optimization problem, p. 55
- `insight_reset_progress`: Reset progress metrics, p. 56
- `insight_results_available`: Initiate early capture of model results, p. 57
- `insight_update`: Update single progress metric, p. 58
- `insight_update_progress`: Update all progress metrics, p. 59
- `insight_use_original_data`: Distinguish model load and run, p. 60
insight_end_initializations

**Purpose**
Indicates that initializations in the model are completed. Replaced with `insightpopulate`.

**Synopsis**
```
procedure insight_end_initializations
```

**Further information**
This function marks the point in the model where the following occurs:

- The model schema and input data is captured when a scenario is loaded.
- The input data is injected when a scenario is run

When the scenario is run in Insight, `insight_end_initializations` behaves the same as `insightpopulate`. When the scenario is loaded, the model terminates at `insight_end_initializations`. When the model is run outside of Xpress Insight, `insight_end_initializations` has no effect.

**Related topics**
- `insightgetmode`
- `insightpopulate`
Deprecated Functionality

insight_minimize, insight_maximize

**Purpose**
Solves the optimization problem defined by the Mosel model. Replaced with insightminimize/insightmaximize

**Synopsis**
- procedure insight_maximize(alg:integer, obj:linctr)
- procedure insight_maximize(obj:linctr)
- procedure insight_minimize(alg:integer, obj:linctr)
- procedure insight_minimize(obj:linctr)

**Arguments**
- alg The solver algorithm choices, as defined by mmxprs.maximize
- obj The objective function constraint

**Related topics**
insightminimize insightmaximize
insight_reset_progress

**Purpose**
Resets the progress state for each progress metric back to zero. Replaced with `insightresetprogress`.

**Synopsis**
```
procedure insight_reset_progress
```

**Further information**
The Xpress Insight system reports a number of metrics during the process of solving the optimization model (see `insightupdate` and `insightupdateprogress`). This procedure sends notifications to reset the value for each metric to zero.

**Related topics**
`insightresetprogress insightupdate insightupdateprogress`
insight_results_available

**Purpose**
Indicate to Xpress Insight that the model results are available to capture by the system. Replaced with `insightresultsready`

**Synopsis**
```plaintext
procedure insight_results_available
```

**Further information**
1. This function can be used to trigger the early capture of the results data. The default behavior is for the system to capture the results data at the end of the model execution.
2. This function can be called from within a Mosel callback for example, `intsol`.
3. If this function is called multiple times, then only the first call will result in results data being captured by the system. Subsequent calls will be ignored.

**Related topics**
`insightresultsready`
insight_update

**Purpose**
Sends a progress update notification for a single metric from the model to the Xpress Insight system. Replaced with `insightupdate`.

**Synopsis**

```plaintext
procedure insight_update(type:integer, value:real)
```

**Argument**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>The type of metric to update.</td>
</tr>
<tr>
<td><code>INSIGHT_GAP</code></td>
<td>The gap to the optimal solution, as a percentage</td>
</tr>
<tr>
<td><code>INSIGHT_OBJVAL</code></td>
<td>The best solution value found so far</td>
</tr>
<tr>
<td><code>INSIGHT_NUMSOLS</code></td>
<td>The count of feasible solutions found so far</td>
</tr>
<tr>
<td><code>INSIGHT_OBJSENSE</code></td>
<td>The direction of the solve. 1 for minimize, -1 for maximize.</td>
</tr>
</tbody>
</table>

**Further information**

1. This function allows the model to report back progress to the system where it is accessible by a client for display. The Analyst Client displays the progress metrics in real time in the Job Queue pane.

2. The Xpress Insight event mechanism does not use the mmjobs send function.

3. By default, mminsight registers the `insightupdateprogress` function provided by the mminsight package as the callback function for the gapnotify callback. The `insightupdateprogress` function calls `insightupdate` for any metric that has changed. It is expected that the developer will only need to call `insightupdate` directly if the gapnotify callback is registered against a function provided by the developer.

**Related topics**

- `insightupdate`
- `insightresetprogress`
- `insightupdateprogress`
insight_update_progress

**Purpose**
Sends update notifications for all supported progress metrics to the Xpress Insight system.
Replaced with `insightupdateprogress`

**Synopsis**
procedure insight_update_progress

**Further information**
1. This procedure calls `insightupdate` for each progress metric supported.
2. By default, mminsight registers the insightupdateprogress function provided by the mminsight package as the callback function for the gapnotify callback. The insightupdateprogress function calls `insightupdate` for any metric that has changed. It is expected that the developer will only need to call insight_update directly if the gapnotify callback is registered against a function provided by the developer.

**Related topics**
`insightupdateprogress insightupdate insightresetprogress`
**insight_use_original_data**

**Purpose**
Indicates whether the model is being loaded or run. Replaced with `insightgetmode`

**Synopsis**
function `insight_use_original_data`: boolean

**Return value**
- true  
  When an Xpress Insight scenario is being loaded, or the model is run outside of Xpress Insight.
- false  
  When an Xpress Insight scenario is being run

**Related topics**
`insightgetmode`
APPENDIX A

Contacting FICO

FICO provides clients with support and services for all our products. Refer to the following sections for more information.

Product support

FICO offers technical support and services ranging from self-help tools to direct assistance with a FICO technical support engineer. Support is available to all clients who have purchased a FICO product and have an active support or maintenance contract. You can find support contact information on the Product Support home page (www.fico.com/support).

On the Product Support home page, you can also register for credentials to log on to FICO Online Support, our web-based support tool to access Product Support 24x7 from anywhere in the world. Using FICO Online Support, you can enter cases online, track them through resolution, find articles in the FICO Knowledge Base, and query known issues.

Please include ‘Xpress’ in the subject line of your support queries.

Product education

FICO Product Education is the principal provider of product training for our clients and partners. Product Education offers instructor-led classroom courses, web-based training, seminars, and training tools for both new user enablement and ongoing performance support. For additional information, visit the Product Education homepage at www.fico.com/en/product-training or email producteducation@fico.com.

Product documentation

FICO continually looks for new ways to improve and enhance the value of the products and services we provide. If you have comments or suggestions regarding how we can improve this documentation, let us know by sending your suggestions to techpubs@fico.com.
Sales and maintenance

**USA, CANADA AND ALL AMERICAS**

*Email:* XpressSalesUS@fico.com

**WORLDWIDE**

*Email:* XpressSalesUK@fico.com

*Tel:* +44 207 940 8718

*Fax:* +44 870 420 3601

Xpress Optimization, FICO

FICO House

International Square

Starley Way

Birmingham B37 7GN

UK

Related services

**Strategy Consulting:** Included in your contract with FICO may be a specified amount of consulting time to assist you in using FICO Optimization Modeler to meet your business needs. Additional consulting time can be arranged by contract.

**Conferences and Seminars:** FICO offers conferences and seminars on our products and services. For announcements concerning these events, go to www.fico.com or contact your FICO account representative.

About FICO

FICO (NYSE:FICO) delivers superior predictive analytics solutions that drive smarter decisions. The company’s groundbreaking use of mathematics to predict consumer behavior has transformed entire industries and revolutionized the way risk is managed and products are marketed. FICO’s innovative solutions include the FICO® Score—the standard measure of consumer credit risk in the United States—along with industry-leading solutions for managing credit accounts, identifying and minimizing the impact of fraud, and customizing consumer offers with pinpoint accuracy. Most of the world’s top banks, as well as leading insurers, retailers, pharmaceutical companies, and government agencies, rely on FICO solutions to accelerate growth, control risk, boost profits, and meet regulatory and competitive demands. FICO also helps millions of individuals manage their personal credit health through www.myfico.com. Learn more at www.fico.com. FICO: Make every decision count™.
insightgetprojattach, 34
insightgetscenattach, 35
insightgetusername, 11
insightiteminfo, 26
insightlistattachtags, 36
insightlistprojattach, 37
insightlistprojattachbytag, 38
insightlistsценattach, 39
insightlistsценattachbytag, 40
insightmaximize, 12
insightminimize, 12
insightpopulate, 13
insightpreparectrs, 14
insightprojattachinfo, 41
insightputscenattach, 42
insightrenamescenattach, 43
insightresetprogress, 15
insightresultsready, 16
insightscenattachinfo, 44
insightsetscenattachdesc, 45
insightsetscenattachhidden, 46
insightsetscenattachtags, 47
insightupdate, 17
insightupdateprogress, 18

L
list attachment tags, 36
list project attachments, 37
list project attachments by tag, 38
list scenario attachments, 39
list scenario attachments by tag, 40

M
maximize, 12
minimize, 12

P
populate inputs, 13
prepare constraints, 14
put scenario attachment, 42

Q
query project attachment, 41
query scenario attachment, 44

R
rename scenario attachment, 43
reset progress metrics, 15
result data
delete, 23
run mode, 10

S
set scenario attachment description, 45
set scenario attachment hidden, 46
set scenario attachment tags, 47
solve optimization problem, 12

U
unit
abbreviation, 24
conversion, 24
conversion factor, 25
conversion offset, 25
definition, 24
factor entity, 24
scale value, 25
update progress metric, 17
update progress metrics, 18

X
xpri_getusername, 19